REMARKS/ARGUMENTS

Claims 1-27 are pending. No claim is amended.

35 U.S.C. 101

The Examiner rejected claims 25-27 under 35 U.S.C. 101. The Applicants respectfully disagree. The Examiner argues the claims are "directed to software per se as the broadest reasonable interpretation for the means for the claimed apparatus includes software means only." The Examiner points to a portion of the specification that states: "For example, various aspects described above may be implemented using firmware, software, or hardware. Aspects of the present invention may be employed with a variety of different file formats, languages, and communication protocols and should not be restricted to the ones mentioned above." Applicants submit that "software per se" is not non-statutory per se.

Applicants respectfully disagree that the subject matter of claims 25-27 are non-statutory. Under the analysis enumerated in MPEP 2106.IV, the claim is properly directed to an apparatus in a means plus function format. Assuming for arguments' sake that the claimed apparatus is directed in an "abstract idea, law of nature, or natural phenomenon," Applicants submit that the claimed invention is a practical application thereof under MPEP 2106.IV.C.2 because it is transformative and/or "otherwise produces a useful, concrete, and tangible result." The claimed invention transforms a top level module and submodules of a design module library to become a test design by "instantiating the I/O structure of a top level module," "selecting a plurality of submodules from a design module library," "parameterizing the plurality of submodules," and "providing logic to interconnect the plurality of parameterized submodules." The module and submodules are transformed to a different state or thing, namely test designs. Furthermore, these means elements provide structural and functional interrelationships between functional elements and the rest of a system such as a computing processing system and provide a "useful, tangible, and concrete result." The result is useful because it generates specifically and substantially a "plurality of test designs having varied characteristics to allow testing of a design automation tool." The result is tangible because real-world test designs are generated. The result is also concrete because the result is "substantially repeatable."

In the response to arguments, the Examiner maintained this rejection "because the specification only discloses generalities regarding the hardware" but not "specific algorithms."

The Federal Circuit ruled that specific algorithms are "not required to produce a listing of source code or a highly detailed description of the algorithm to be used to achieve the claimed functions." Aristocrat Technologies Australia v. International Gaming Technologies. Sufficient structure is found if the specification "at least disclose[s] the algorithm that transforms the general purpose microprocessor to a "special purpose computer programmed to perform the disclosed algorithm." Id. Applicants submit that Figure 5, 6, 7, and 8 and associated description disclosed in the present specification describes an algorithm "that transforms the general purpose microprocessor to a special purpose computer programmed to perform the disclosed algorithm." Applicants submit that the Figures 5, 6, 7, and 8 and associated description discloses just as much or more specificity than that of the patent in dispute in WMS Gaming, where the Federal Circuit found "the structure disclosed for the "means for assigning" limitation of claim 1 of the Telnaes patent is a microprocessor programmed to perform the algorithm illustrated in Figure 6." Thus, the algorithm disclosed in the present specification the "specific algorithm" required to satisfy the Federal Circuit."

If the Examiner continues to contend that the algorithm disclosed in the present specification is deficient in someway as the specific algorithm that transforms the general purpose microprocessor to a special purpose computer programmed to perform the disclosed algorithm, Applicants respectfully request that the Examiner state the deficiencies. For at least the above reasons, withdrawal of this rejection is respectfully requested.

35 U.S.C. 103

Claims 1-2, 5, 17-21, and 25-27 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten (U.S. Patent No. 5,805,795) in view of Bening et al. ("Optimizing Multiple EDA Tools within the ASIC Design Flow"). Claims 4, 6-13, 15, and 22-24 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in further view of Zaidi (2002/0038401 A1). Claims 14-16 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in further view of Zaidi and Goossens ("Design of Heterogeneous ICs for Mobile and Personal Communication Systems"). Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in further view of Rajsuman (U.S. Patent No. 6,678,645). Applicants respectfully traverse.

Whitten is directed to a "method for selecting a set of test cases which may be used to test a software program product . . . The program to be tested may have a number of code blocks that

may be exercised during execution of the program. The method includes identifying each of the code blocks that may be exercised, and determining a time for executing each of the test cases in the set. A set of the test cases is then selected that exercises a maximum number of the identified code blocks that can be exercised in a minimum time." (Abstract) The Background section discusses how a "suitable test for a particular software product" is developed:

> a software test designer studies the product specifications, extracts a list of product features, and generates a number of "assertions" regarding the software product. An assertion is, for example, a specification of the behavior of the program, or part thereof, when it is operating properly, a statement of how the program is expected to operate under certain conditions, or other appropriate performance or result oriented product expectation.
>
> For each assertion, the designer then develops a set of "test cases" in the form of software that exercises as many as possible of the code blocks in the product to prove the validity of the assertions. Typically, a designer may generate hundreds, even thousands, of test cases for a typical complex software product, so choosing a particular set of test cases to apply can be difficult, particularly if an optimal, or nearly optimal, set of test cases is sought to be generated. (The term "optimal" is used herein to mean that the test coverage is maximized across a preselected number of product code blocks in a minimized preselected amount of time.) (column 1, lines 26-44)

The quoted passage describes a software test designer "generat[ing] hundreds, even thousands, of test cases for a typical complex software product" to test "the validity of the assertions" and that it is difficult to "choos[e] a particular set of test cases to apply ... particularly if an optimal ... set ... is sought." Together with the Abstract, it is clear that the Whitten method seeks to produce this optimal "set of test cases" from the "hundreds, even thousands, of test cases" generated by a software test designer.

The Examiner relies on Whitten to describe "generating a plurality of test designs." The Examiner cites column 8, lines 15-21 as support for "generating a plurality of test designs." The passage states:

> After the iterative process has been completed, a string is derived for use as the final population which has the highest fitness value. This string is interpreted, as described above with reference to FIG. 2. Once the set of test cases has been determined, it may be used to achieve a good tradeoff between test coverage and test execution time on any target operating system and hardware.

Applicants submit that this passage does not support "generating a plurality of test designs." Whitten discloses selecting and determining a set of test cases to apply for testing from the "hundreds, even thousands, of test cases" created by a software test designer. Determining "the set of test cases" in this context is not "generating a plurality of test designs" in the claimed invention because the "set of test cases" are determined (selected) from a collection of existing test cases and the "plurality of test designs" is generated.

The Examiner also relies on Whitten to describe "selecting a plurality of submodules from a design module library" and "wherein a probabilististic function is applied to select submodules of different types from the library." The Examiner responded to Applicants' previous argument that "the blocks of code executed by the test cases, as recited in column 5, lines 22-52, are submodules." Applicants respectfully disagree.

The specification explicitly describes submodules. "Possible submodules include memory modules 431 and 433, phase locked loop module 451, adder 453, filter 455, and timer 457. The top-level module 401 also includes registers 441, 443, 445, and 447. In this example, other submodules included in the top-level module 401 are DSP core 461, processor core 463, etc. Each submodule may also be associated with various input and output lines. The various input and output lines can be categorized as clock lines, control lines, or data lines." (page 12, lines 25-31). These submodules are design modules selected "from a design module library" to generate "a plurality of test designs."

The "code blocks" tested by the test cases of Whitten are not submodules. These code blocks are a part of the product program to be tested. Column 5, lines 22-52 of Whitten describes using a compiler to identify "all of the code blocks in the program 12 to be tested " by "generat[ing] a list 16 of code blocks contained in the product source code as well as an instrumented version of the product." These code blocks are a part of the software program to be tested, and not a part of the test cases.

Furthermore, these code blocks are not "selected" "from a design module library." The Examiner may have associated the collection of test cases "initially generated or collected by a test designer" (Whitten, column 4, line 65-66) to a "design module library." However, this association is incorrect because test cases are not submodules. The example submodules form the specification includes memory modules, phase locked loop module, adder, filter, timer,

registers, DSP core, and processor core. (page 12, lines 25-31) Test cases designed to test a software program product are not any of these.

Dependent claims 7 and 8 recite submodule input lines and submodule output lines. If submodule is interpreted to mean test case, submodule input lines and submodule output lines would make no sense. Whitten software generated test cases do not have input lines or output lines.

Bening does not cure the defects of Whitten. Bening discloses a "proposed design methodology" to "let engineers optimize a design's functionality and simultaneously support multiple EDA tools." (Abstract, page 46) Bening is about a design methodology and not about EDA tool testing or generating test designs. The Examiner points to various portions of Bening as disclosing various claim elements that are not supported by Bening. For example, the Examiner points to Bening at page 47 as disclosing "creating a test design for an EDA tool comprising instantiating the I/O structure of a top level module." The cited passage states "Verilog, on the other hand, provides a mechanism—a module—for creating abstraction in the design. The module provides the details for the abstracted interface as well as the abstracted behavior. Verilog can link a different definition for a specific, instantiated module by referencing application-oriented libraries. With this technique, engineers can generate optimized libraries for specific process points in the design flow." This cited passage does not teach "instantiating the I/O structure of a top level module." Instead, it teaches that "Verilog can link a different definition for a specific, instantiated module by referencing application-oriented libraries." Applicants believe that linking "a different definition for a specific, instantiated module" does not teach one skilled in the art to "instantiating the I/O structure of a top level module."

The Examiner concluded that it would not have been "obvious to an ordinary person skilled in the art to "combined the method a generating a plurality of test designs of Whitten with the creation of test designs with EDA tools of Bening in order to optimize EDA tool." As explained above, Whitten does not teach a method of "generating a plurality of test designs." Whitten discloses only a method of selecting which of the generated test cases to test. Bening does not teach creating test designs with EDA tools—Bening teaches a design methodology with which "engineers can focus on design functionality rather than on each individual electronic design automation (EDA) tool's optimal coding style requirement." (page 46, 1st paragraph) In

other words, Bening seeks a design methodology that does not depend on individual EDA tools. Thus, one skilled in the art would not have combined these two references. As explained above, even if one had, the combined result would not have been the claimed invention.

Claims 3-4, 6-13, 14-16, and 22-24 were rejected in further combination with other references, Zaidi, Goossens, and Rajsuman. None of these references cure the deficiencies of Whitten and Whitten with Bening. For at least the foregoing reasons, withdrawal of this rejection is respectfully requested.

In light of the above remarks, the rejections to the independent claims are believed overcome for at least the reasons noted above. Applicants believe that all pending claims are allowable in their present form. Please feel free to contact the undersigned at the number provided below if there are any questions, concerns, or remaining issues.

Respectfully submitted,
Weaver Austin Villeneuve & Sampson LLP

/Cindy H. Shu/

Cindy H. Shu
Reg. No. 48,721

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100